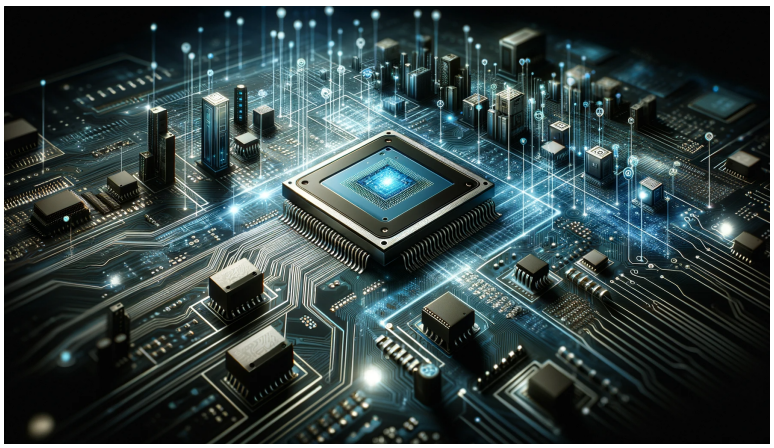
8-BIT SUPER REGISTER

EXPLORING THE PINNACLE OF DIGITAL LOGIC DESIGN THROUGH THE VERSATILITY OF THE 8-BIT SUPER REGISTER

EDITED BY

MUHAMMED IBRAHIM RISK 21011047
AHMED SAEED MUHAMMED 21010091
YAHYA EMAD EL-DEEN EL-KONY 21011557
YEHIA SAID GEWILY 21010705

University of Alexandria



EEC-242 LOGIC CIRCUITS DESIGN

DEC 2023

LaTeX

Introduction

*The **Super Register** is an innovative component that excels in data handling, offering functionalities as both a shift register and a counter. As a shift register, it is adept at loading data in parallel and carrying out shifting or rotation in both directions with finesse. When functioning as a counter, it efficiently performs incremental or decremental counting. Equipped with eight distinct operations, the Super Register allows for loading new data, shifting, rotating, data storage, and counting capabilities, making it a versatile and indispensable element in data processing.*

The code is defined by a set of inputs that govern its operations. The clock input (clk) signals the start of operations in sync with the clock signal's rising edge. The 8-bit data input (data_in) is crucial for introducing new data to the register. The control input (control), consisting of 3 bits, directs the selection of the Super Register's operations. The shift-in bits (data_sh_r and data_sh_l), both single-bit inputs, are integral during right and left shifts. The outputs of the Super Register are twofold: the primary 8-bit output delivers the processed data, while the secondary single-bit output indicates a special condition where it becomes '1' if the register's output is all zeros or all ones, and '0' in all other cases.

- ## *Functionality*

The Super Register serves multiple purposes. It has the capability to load data words simultaneously and execute shifts or rotations in any direction. Additionally, it functions as a counter with the ability to increment or decrement counts.

- ## *Inputs and Outputs*

i. The Super Register is equipped with five inputs for its operation:

a) ***clk:*** *This is the clock input that triggers the commencement of operations with the ascending transition of the clock pulse.*

b) ***data_in:*** *An 8-bit input that serves to introduce new data into the register.*

c) ***control:*** *A 3-bit input that selects from among the eight possible operations that the Super Register can execute.*

d) ***data_sh_r & data_sh_l:*** *These are individual 1-bit inputs that are used to feed in shift data when the register is shifting to the right or left, respectively.*

ii. The Super Register features two outputs:

a) *The primary output is an 8-bit value reflecting the data that has been processed and is exiting the register.*
b) *The secondary output is a single bit that indicates a specific condition: it is set to '1' if the register's output is a binary sequence of either all zeros ("00000000") or all ones ("11111111"); in all other scenarios, this output is '0'.*

- ## *Some Definitions:*

I. ***Right Shifting****: Moves bits in a data word to the right by one position, with the leftmost bit typically filled with zero.*

II. ***Left Shifting:*** *Shifts bits in a data word to the left by one position, filling the rightmost bit with zero.*

III. ***Right Rotating:*** *Circulates bits in the data word to the right, with the rightmost bit wrapping around to the leftmost position.*

IV. ***Left Rotating:*** *Circulates bits in the data word to the left, with the leftmost bit moving to the rightmost position.*

V. ***Data Storing:*** *Maintains the current data state without any changes.*

VI. ***New Data Loading:*** *Replaces the current data in the register with a new data word.*

VII. ***Counting Up:*** *Increments the data value, typically adding one each time.*

VIII. ***Counting Down:*** *Decrements the data value, subtracting one each time.*

VHDL Code

- ***Library***

```
1       --@@@@@@  BEGIN LIBRARY  @@@@@@--
2   LIBRARY IEEE;
3   USE IEEE.std_logic_1164.all;
4   USE IEEE.std_logic_arith.all;
5   USE IEEE.std_logic_unsigned.all;
6       --@@@@@@  END LIBRARY  @@@@@@--
```

The IEEE library is a critical component in VHDL, providing essential logic types and arithmetic functions. It encompasses the std_logic_1164 package, which supports elementary logic operations, the std_logic_arith package for arithmetic functionalities, and the std_logic_unsigned package for operations with unsigned numbers.

- ***Entity***

```
8       --@@@@@@  BEGIN ENTITY  @@@@@@--
9
10  ENTITY SuperRegister IS                                                 --      @@@  Control i/p  @@@
11      PORT(data_in        :in     std_logic_vector(7 downto 0);           --      000  ~>  LOAD NEW DATA
12           control        :in     std_logic_vector(2 downto 0);           --      001  ~>  SHIFT RIGHT
13           clk            :in                      std_logic;             --      010  ~>  SHIFT LEFT
14           data_sh_r      :in                      std_logic;             --      011  ~>  ROTATE RIGHT
15           data_sh_l      :in                      std_logic;             --      100  ~>  ROTATE LEFT
16           data_out       :out    std_logic_vector(7 downto 0);           --      101  ~>  STORE PRESENT DATA
17           Z              :out                     std_logic);            --      110  ~>  COUNT UP
18  END SuperRegister;                                                      --      111  ~>  COUNT DOWN
19
20      --@@@@@@  END ENTITY  @@@@@@--
```

*The **Super Register** entity is a multifunctional register that can operate both as a universal shift register and as a counter. It is designed to execute eight different operations, which are determined by the specific control inputs provided.*

- ***Inputs:***

1. data_in: *This is an 8-bit vector that is used to input new data into the register.*

2. control: *This 3-bit vector determines which operation the register will execute.*

3. clk: *This clock input triggers the start of operations at its rising edge.*

4. data_sh_r: *This input is used to control the rightward shift of the register.*

5. data_sh_l: *This input manages the leftward shift within the register.*

- ***Outputs:***

1. data_out: *An 8-bit vector that reflects the data being outputted from the register.*

2. Z: *A single-bit output that signals specific states, like a full set of ones or zeroes in the register's output.*

- ***Architecture***

a) Starting with declaring the signals

```
24  ARCHITECTURE BEHAVIOR OF SuperRegister IS
25      SIGNAL    count:            std_logic_vector(7 downto 0):=(OTHERS => '0');
26      SIGNAL    store:            std_logic_vector(7 downto 0)                 ;
27      SIGNAL reg_data:            std_logic_vector(7 DOWNTO 0):=(OTHERS => '0');
```

b) ***Count Signal:*** *This signal is utilized to control the counting function, enabling the register to count either upwards or downwards.*

c) ***Reg_data****: This is employed to retain the latest data resulting from shifts or rotations, holding it in preparation for subsequent operations.*

d) ***Store Signal:*** *This signal is dedicated to preserving data from any executed operation, ensuring it is available for output retrieval.*

- ***Super Register Behaviour:***

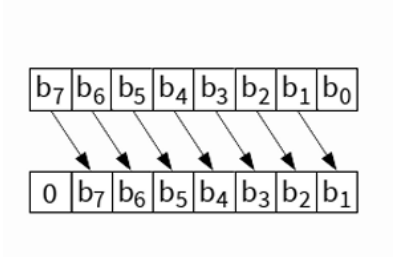
```
29  BEGIN
30
31      PROCESS(clk)
32      BEGIN
33          IF rising_edge(clk) THEN
34
35              CASE control IS
36
37                  WHEN "000"      =>              ------------------------------- LOAD NEW DATA
38                      store       <=  data_in   ;
39                      reg_data    <=  data_in   ;
40
41                  WHEN "001"      =>              ------------------------------- SHIFT RIGHT
42                      reg_data    <=  data_sh_r&reg_data(7 downto 1)  ;
43                      store       <=  data_sh_r&reg_data(7 downto 1)  ;
44
45                  WHEN "010"      =>              ------------------------------- SHIFT LEFT
46                      reg_data    <=  reg_data(6 downto 0)&data_sh_l  ;
47                      store       <=  reg_data(6 downto 0)&data_sh_l  ;
48
49                  WHEN "011"      =>              ------------------------------- ROTATE RIGHT
50                      reg_data    <=  reg_data(0)&reg_data(7 downto 1) ;
51                      store       <=  reg_data(0)&reg_data(7 downto 1)  ;
52
53                  WHEN "100"      =>              ------------------------------- ROTATE LEFT
54                      reg_data    <=  reg_data(6 downto 0)&reg_data(7) ;
55                      store       <=  reg_data(6 downto 0)&reg_data(7)  ;
56
57                  WHEN "110"      =>              ------------------------------- COUNTING UP
58                      count       <=  count + 1 ;
59                      store       <=  count     ;
60
61                  WHEN "111"      =>              ------------------------------- COUNTING DOWN
62                      count       <=  count - 1 ;
63                      store       <=  count     ;
64
65                  WHEN OTHERS     =>              ------------------------------- STORE PRESENT DATA
66                      store       <=  store     ;
67
68              END CASE;
69
70          END IF ;
71      END PROCESS ;
```

a) *Upon encountering a rising edge in the clock signal, the VHDL code initiates its operation sequence.*

b) *The operations are selected by a CASE statement, which responds to the 'Control' parameter's value.*

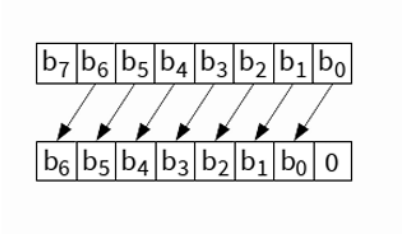
c) *When **'Control'** is set to 000 in the VHDL code, it triggers the loading of new data from the data_in input.*

d) *If **'Control'** equals 001 the VHDL code executes a rightward shift of the data, storing the shifted result in reg_data.*



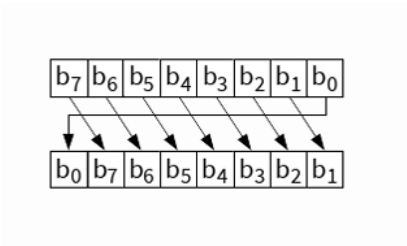
Shift Right

e) *When the Control parameter equals "010" in the VHDL code, it performs a leftward data, and then stores the shifted data into reg_data.*



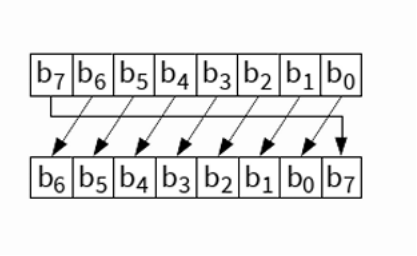
Shift Left

f) *when the Control parameter is assigned the value "011," it initiates a rotation of the data to the right, with the rotated data being subsequently saved in reg_data.*



Rotate Right

g) *If the Control is set to "100" in the VHDL code, it triggers a leftward rotation of the data, followed by storing this rotated data in reg_data.*



Rotate Left

h) *When the Control is set to "101" in the VHDL code, it maintains the existing data state without making any changes.*

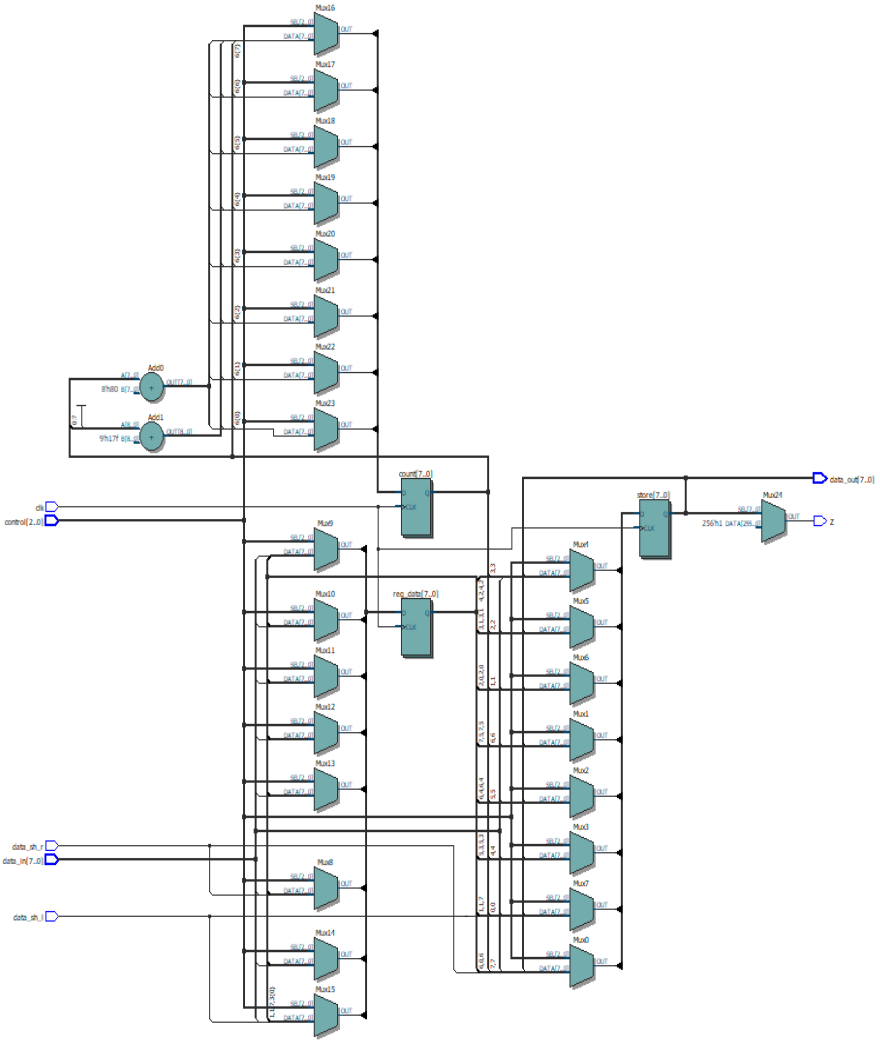
i) *If the Control is assigned "110," the VHDL code initiates a count-up operation, ranging from 0 to 255.*

j) *With the Control value at "111," the VHDL code starts a count-down operation from 255 to 0.*

k) *Finally, the store signal captures the value resulting from a specific operation, which is then sent to the output. Additionally, the output Z is designed to check if the output is entirely ones or zeroes.*

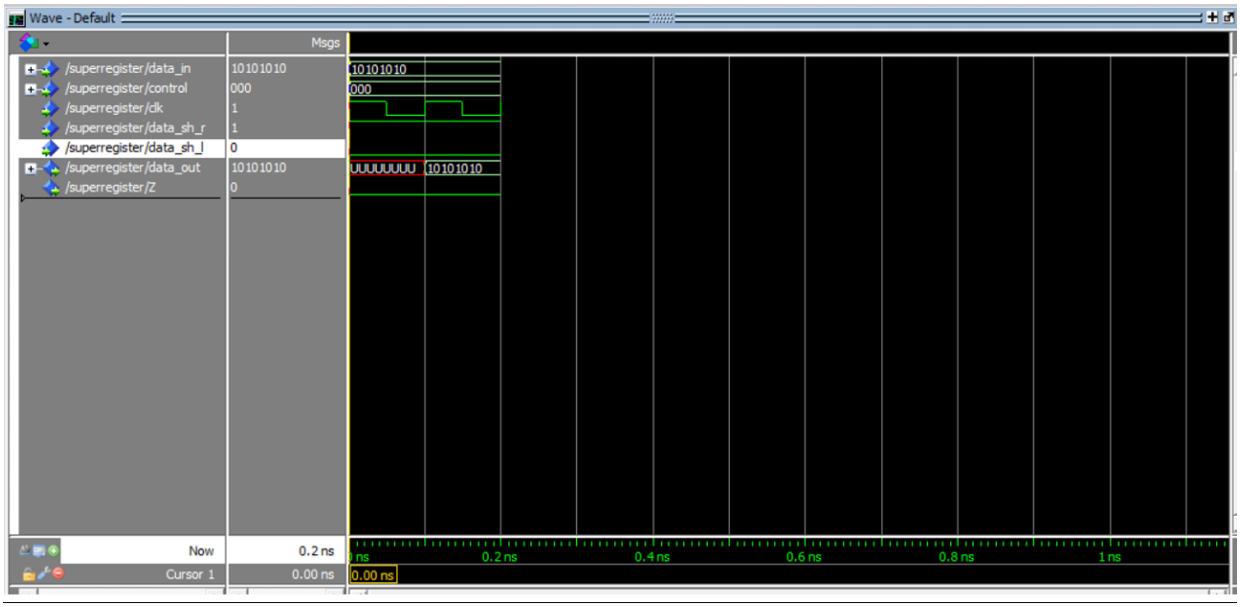
```
73    PROCESS(store)
74    BEGIN
75
76            data_out <= store;
77
78    END PROCESS;
79
80            -- SECOND OUTPUT THAT CHECKS ALL ONE'S OR ALL ZEROES' --
81       WITH store SELECT
82       Z <= '1' WHEN "11111111",
83            '1' WHEN "00000000",
84            '0' WHEN OTHERS;
85
86  END BEHAVIOR;
87                  --@@@@@@  END ARCHITECTURE  @@@@@@--
```

Circuit Schematic:

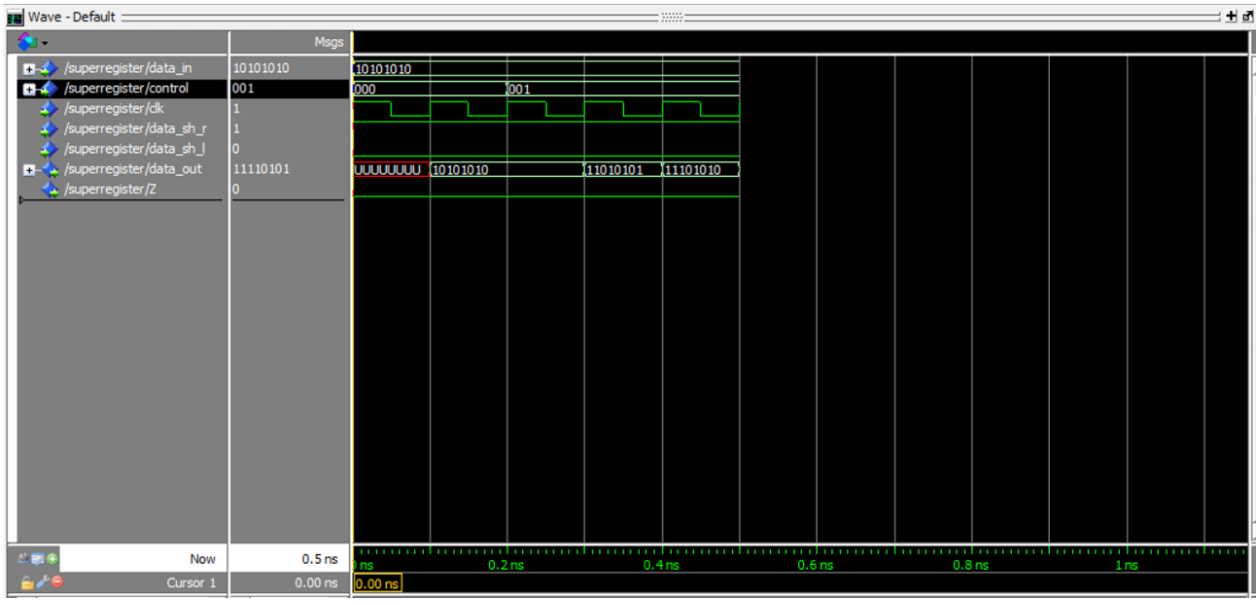


ModelSim Simulation

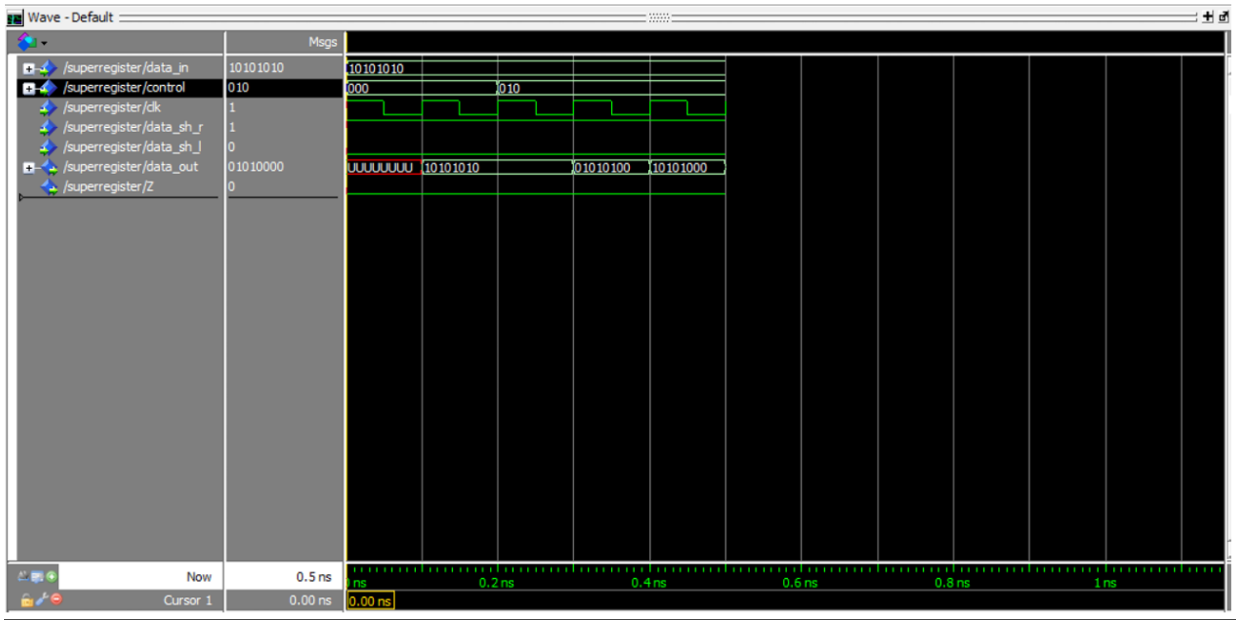
- ***Loading New Data***



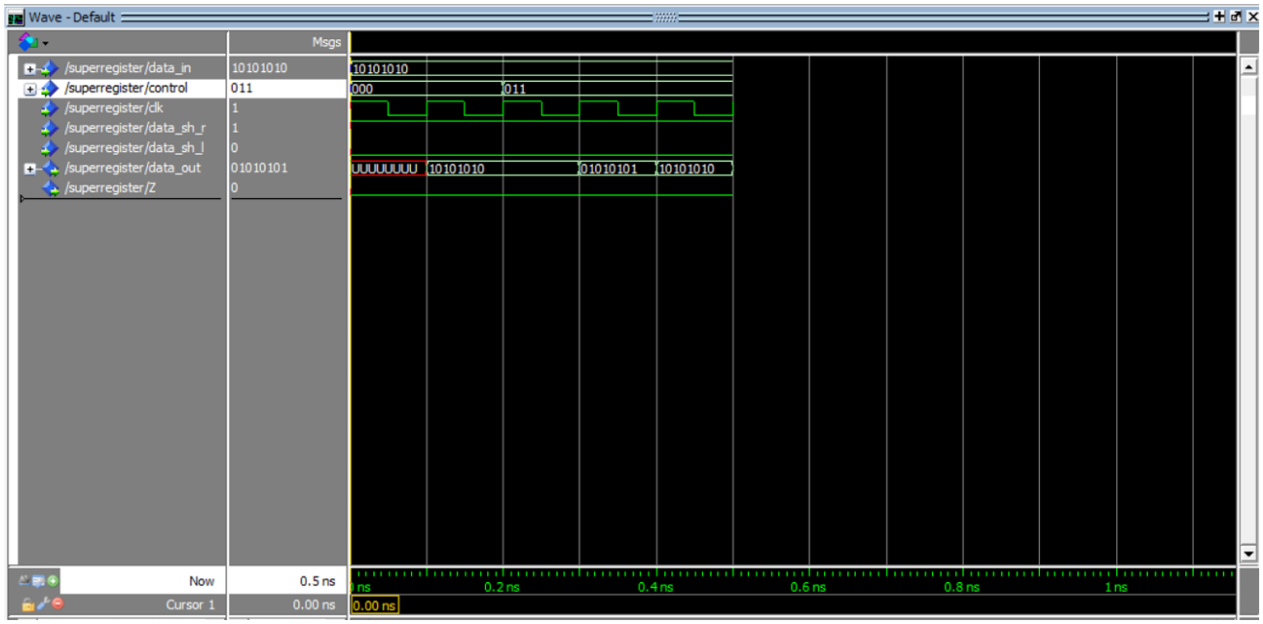
- ***Right Shifting***



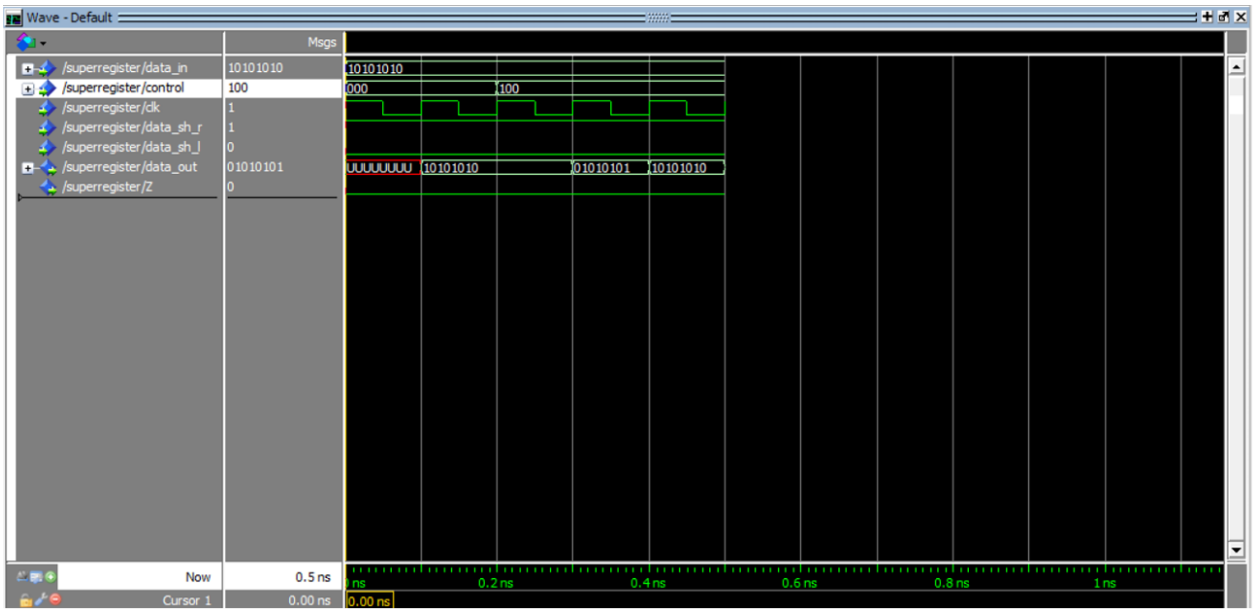
- ***Left Shifting***



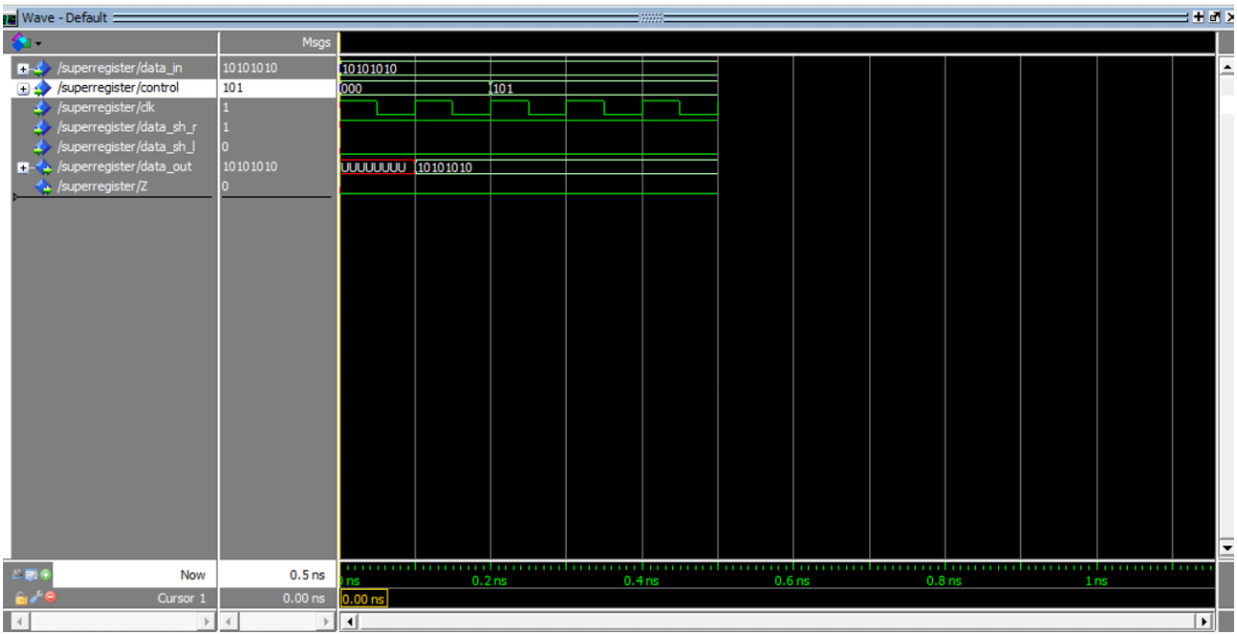
- ***Right Rotating***



- ***Left Rotating***

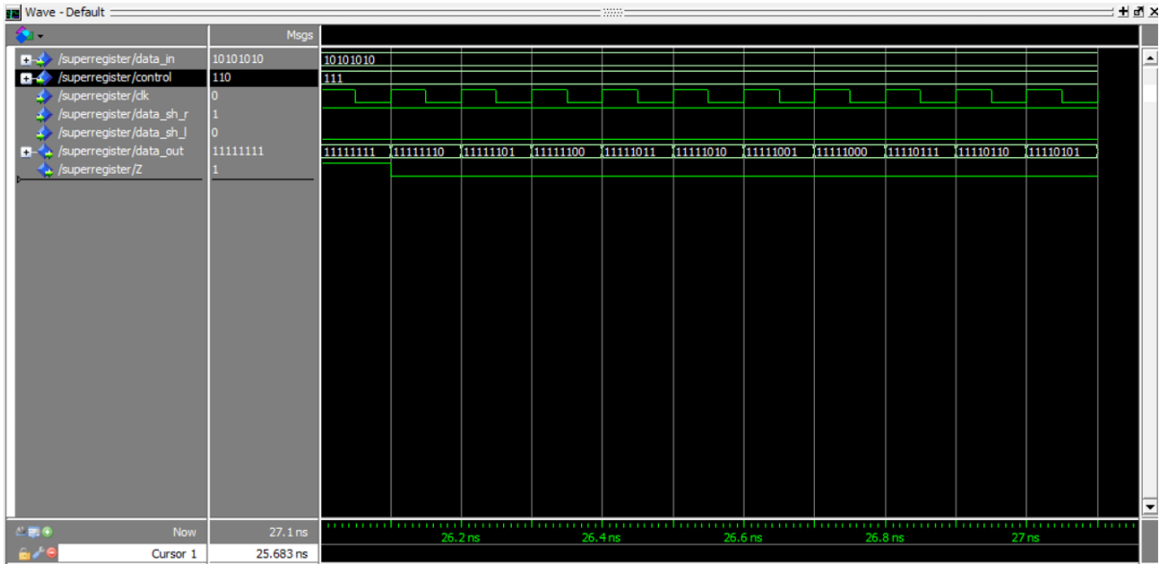


- ***Present Data Storing***



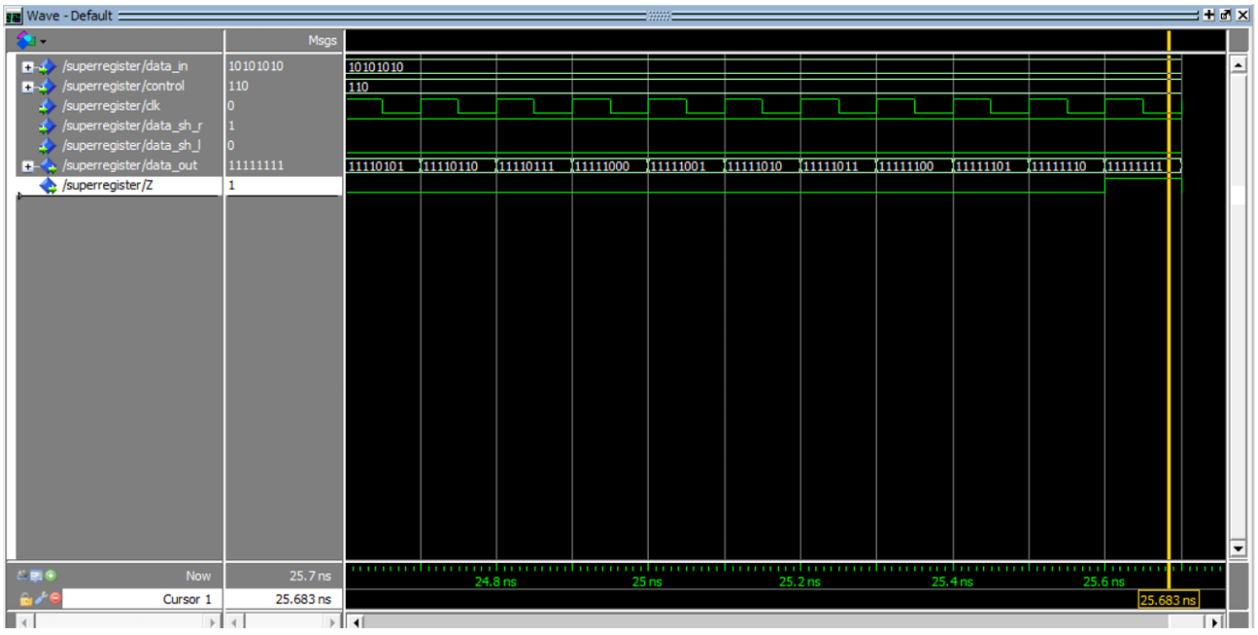
- ***Counting Up:***

- ***Counting Down:***



- ***Load → Right Shifting→ Left Rotating → Left Shifting → Data Storing → Right Rotating.***

- ***Z output:***



Conclusion:

*The **Super Register** distinguishes itself as a versatile and efficient tool in the field of data manipulation, skilfully combining the functions of both a universal register and a counter. With its capacity to execute eight distinct operations, it boasts a robust input system and offers dual modes of output, making it a crucial component in various applications. Its functionalities, ranging from data loading, shifting, and rotating, to storing and counting, ensure that it can adeptly handle diverse tasks. The precision of its clock input guarantees that operations are perfectly timed, while its specialized outputs deliver insightful information about the register's status. In sum, the Super Register's combination of flexibility, precision, and reliability makes it an invaluable resource in digital design and data management.*